DOCUMENT RESUME

ED 222 196 IR 010 432

AUTHOR 3ates, Madeleine; And Others
TITLE Generative Tutorial Systems.
PUB DATE Mar 81
NOTE 18p.; Paper presented at the Annual Meeting of the Association for the Development of Computer-Based Instructional Systems (Atlanta, GA, March 3-5, 1981).

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Computational Linguistics; *Computer Assisted Instruction; *Computer Programs; *English Instruction; Individual Needs; Microcomputers; *Programed Tutoring; *Transformational Generative Grammar
IDENTIFIERS Computer Assisted Language Instruction; *Generative Computer Assisted Instruction

ABSTRACT

The advantages of a knowledge-based computer program that creates exercises for language instruction rather than presenting pre-stored materials are discussed, and one such system, ILIAD, is described. Designed to address a broad range of language learning needs, this system can be used by people learning English as a second language and by children or adults with language-delaying handicaps, such as deafness, as well as by native English speakers. The generality and power of the approach are stressed, with emphasis on the computer's ability to meet individual student needs, and its applicability to a wider variety of tutorial purposes and material than traditional computer-assisted instruction systems. Specific examples of several ILIAD tutorials demonstrate the power of a generative system with user-specifiable options, and approaches to computer-assisted language instruction in both traditional and generative systems are presented in tabular form for comparison. Developed on a large computer (InterLisp on a DEC System 20), ILIAD is being re-implemented for microcomputer (UCSD-Pascal on Apple and Cromemco systems). Appendices include sample base and surface structures, as well as sample sentences generated by ILIAD. Sixteen references are listed. (LMM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED222196

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.
Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

GENERATIVE TUTORIAL SYSTEMS

by

Madeleine Bates
Jack Beinashowitz
Robert Ingria
Kirk Wilson

IR010432

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

M. Bates

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

GENERATIVE TUTORIAL SYSTEMS

Madeleine Bates
Bolt Beranek and Newman Inc.
Cambridge, Mass.

Jack Beinashowitz, Robert Ingria, Kirk Wilson
The ILIAD Project
Boston University

Abstract

This paper presents a justification for the generative approach in computer assisted instruction. We discuss the advantages of a knowledge-based computer program that creates exercises for language instruction rather than presenting pre-stored material. This approach requires that the computer have knowledge of the material being presented. Using this knowledge in a variety of ways increases the computer's ability to meet the individual needs of the student. Another advantage of having a generative "core" system is that it can be used for a much wider variety of tutorial purposes and over a much broader range of material than traditional CAI systems.[1]

1. Introduction

In the past, computers have been used in education as automatic page turners for "programmed instruction," as reliable file clerks for information retrieval and record keeping, and as big arithmetic calculators; seldom have they been employed as truly knowledgeable entities that can manipulate highly structured symbolic data in highly complex ways to bring some degree of intelligence to the educational environment. Computer assisted instruction systems that actually *use* the knowledge they contain (instead of simply presenting pre-stored text to the user) have enormous educational potential that has seldom been tapped because doing so previously required the resources of a large, expensive computer system.

Dramatic advances in computer technology and information science have made possible many changes in the ways that computers can be used to aid learning. One of the recent effects of this change can be seen in the microcomputer revolution, which is the first step in bringing to a single user the computational power which had previously been available only to those few users having access to large timeshared systems (the recent IBM announcement of a prototype single chip 370-158 is an example of such developments). Two of the more far-sighted researchers in the field of computer-assisted instruction have written the following:

---

[1]This paper was presented at the 1981 Annual Meeting of the Association for the Development of Computer-Based Instructional Systems, Atlanta, Georgia, March 3-5.

Traditional computer-assisted instruction (CAI) paradigms were developed under the assumption that computational power is a scarce resource, and these paradigms are, for the most part, incapable of exploiting the latest technological advances. To effectively use the increased computational power requires a re-evaluation of the role of the computer in instructional paradigms. [5]

2. The Power of a Generative System

In this paper we discuss one particular system, called ILIAD, which has been under development for two years on a large computer (InterLisp on a DEC System 20) and which is currently being re-implemented for a microcomputer (UCSD-Pascal on Apple and Cromemco systems). The goal of ILIAD is to address a broad range of language learning needs, ranging from instruction in basic language skills to help with some of the more complex structures of English. The system can be used by people who are learning English as a second language and by children or adults with various language-delaying handicaps such as deafness as well as by native English speakers. The goal of this paper is not to explain the details of this language tutoring process but to illustrate the generality and power of the approach we have taken.

ILIAD has three primary strengths that make it substantially different from other CAI systems:

- The examples and exercises that are presented to the student are not written by a teacher and pre-stored in the computer, but are *generated* by the computer from a dictionary and a grammar of English.

- The vocabulary level, complexity, and the type and content of the exercises generated by the system are controlled by the *student*.

- The computer system "knows" a great deal about the structure of the sentences it generates and is able to use this knowledge in a wide variety of ways to help the student.

Other systems [1, 6, 8, 9, 12, 13, 15, 14, 16]have attempted to substitute interchangeable words or combine pre-stored sets of sentence fragments to produce sentences, but this method is limited to rather simple variations on a few basic patterns. ILIAD can generate far more types of sentences because its grammar is more effective and powerful than a simple combining algorithm. ILIAD can generate a potentially infinite number of different sentences.

Traditional CAI systems in which the exercises are pre-stored take a long time to develop, are difficult to tailor to the individual needs of a wide range of students, and cannot easily be used as the basis for teaching related topics. In addition, a student who has a great deal of trouble with a particular topic can quickly exhaust the resources of the traditional system and must then either quit or repeat the same exercises. A generative system such as ILIAD also takes a long time to develop, but once the generative mechanism is complete, tutorials that use it are easy to develop, and students have a great deal of control when using the tutorials. Generative tutorials constantly produce new material even when used repeatedly by a single student.

The sentence generation component of ILIAD is based on the linguistic paradigm of transformational grammar and is combined with techniques from computational linguistics such as semantic networks [7] to ensure that the sentences that are produced are semantically meaningful as well as syntactically correct. For each exercise, the type of sentence generated depends upon a large number of *constraints* (for example, the presence/absence of a direct object, the tense of the main clause, and whether the sentence is to be declarative or one of a variety of question types). Some of these constraints can be set by the user in response to a series of questions, some of them are set by the tutorial, and some are set at random by the sentence generator. To avoid confusing beginning students with sophisticated constraint choices, the system asks the student to choose one of four levels of difficulty; this choice governs the choices that are subsequently presented to the student. The user also has control over the level of the vocabulary: this means that a user working on complex sentences can request a simple vocabulary to facilitate understanding, and can raise the vocabulary level as the syntactic issues are mastered.

Once the constraints are set, the system uses them to produce a *base structure* that embodies the basic syntactic relationships (e.g., subject, direct object) and parts of speech (e.g., predicate adjective, relative clause). Then the transformational component of the system operates on the base structure using the constraints to direct the production of one or more *surface structures* that show the relationships of the words in the final sentence(s). Appendix 1 shows some of the information contained in base and surface structures for a simple sentence. Using a simple vocabulary level, the base structure for the first sentence below could be transformed into a number of different sentences including the next six.

Bill ate the cake.
Did Bill eat the cake?
It was eaten by Bill.
Bill should eat the cake.
Who ate the cake?
Didn't he eat it?
It might have been eaten.

With a more complex vocabulary and the same constraints the sentences might be as follows:

The teacher criticized the janitor.
Did the teacher criticize the janitor?
He was criticized by the teacher.
The teacher should criticize the janitor.
Who criticized the janitor.
Didn't she criticize him?
He might have been criticized by the teacher.

This type of generative mechanism can produce literally millions of different sentences from a dictionary of a few hundred words and a few hundred transformational rules. Appendix 2 gives examples of just a few of the types of sentences that ILIAD can generate. Groups of related sentences can be produced from

the same base structure. Since ILIAD knows about those relationships, it can take advantage of this knowledge when interacting with a student.

3. Examples of Knowledge Use

In this section, we will give examples from several ILIAD tutorials which demonstrate the power of a generative system with user-specifiable options. Each tutorial is a small computer program that directs the sentence generation mechanism of ILIAD to create sentences which are appropriate to the subject of the tutorial and then interacts with the user to present exercises, pose problems, get answers, check them, respond to the user, etc. Each tutorial took only a few weeks to develop into prototype form, and although each one calls the same sentence generation mechanism, the type of sentences produced are specific to the instructional topic of each tutorial. The tutorials are not structured in a developmental sequence but are intended to be used independently by users with widely varying needs. In this sense, ILIAD can be viewed as a collection of CAI programs all of which use a powerful common subroutine to generate sentences. Figure 1 illustrates this concept.

ILIAD has a great deal of knowledge about a number of aspects of the sentences it generates, their structure, and their parts. For example, it uses information about the meanings of words in its dictionary to ensure that the sentences it generates are meaningful, not like "The cake saw the lazy idea." ILIAD also knows the part of speech of each of the words in the base structure, the syntactic phrase structure which those categories form (e.g. noun phrase, prepositional phrase, relative clause), relationships among the phrasal constituents (e.g. the noun an adjective modifies), the syntactic roles the phrases fill (e.g. subject, main verb, first noun phrase) in both the base and surface structures, the tense and number of all verbs, the person and number of all nouns (and noun phrases), and the gender and case of nouns where relevant. It also knows the correct punctuation and capitalization of the sentence, and, because only root forms of regular words are stored in the dictionary, it can pluralize nouns, make adjectives comparative and superlative, and inflect (conjugate) verbs. When a sentence is generated, the sentence type (e.g. transitive, intransitive, predicate nominal) is known, and when related sentences are generated from the same base structure the relationship is also known (e.g. negative/affirmative, declarative/interrogative, complete-nouns/pronouns, active/passive). Now we will see how this knowledge is used in some of the tutorials that have been written around the ILIAD generative core.

The Singular/Plural Tutorial uses knowledge of the singular and plural forms of nouns and verbs (irregular as well as regular) and of the syntactic form of sentences to teach students about subject-verb agreement.

The student is presented with a sentence the subject of which is either singular or plural and is instructed to change the subject into the opposite number. This change often necessitates changing the form of the verb as well to agree with the new number

of the subject. An example of a typical interaction will clarify how this tutorial operates[2]:

The doctor helped Dianne.
Now type it with a plural subject:
>*The doctors helped Dianne.*
Right!

If the user had chosen higher vocabulary and syntactic levels, the exercise might have been as follows:

The sad adult was humiliated by the girls.
Type the plural form:
>*The sad adults were humiliated by the girls.*
You got it!

If the user encounters difficulties, a HINT or HELP may be requested as follows:

The stupid child has been making up the questions.
Now type it with a plural subject:
>*hint*
The subject is: **the stupid child.**
>*help*
The plural of "**child**" is **irregular!**
>*The stupid children has*
{The system recognizes the student's error and
retypes the correct words to give another chance.}
>**The stupid children**
>*hint*
The verb "**has been making up**" may also have to be changed.
>*The stupid children have been making up the questions.*
That's right!

In addition to knowing about the forms of English verbs and various syntactic rules, ILIAD also knows about different sentence types. It is a relatively easy task to add to the set of sentence types that the system can produce. Another tutorial was designed upon the request of teachers at the Boston School for the Deaf and uses the patterns that are standard in their curriculum. In this tutorial, ILIAD tests the student's knowledge by presenting a sentence which must be identified as one of the following five sentence types: intransitive, transitive, predicate adjective, predicate nominal, predicate adverb. If the student errs, or asks for a hint, ILIAD shows part of the sentence structure.

---

[2] In all of the examples given in this section, text presented in **bold type** was generated by ILIAD's sentence generator; that in regular type was printed by the tutorial program, including the symbol > which is a prompt to the student to type something. *Italic text* was elicited from the student; in many cases abbreviations and shortcuts have been developed to minimize the amount of actual typing the user has to do, but this is not indicated in the transcripts.

The foolish soldier doesn't seem happy.
What type is that?
>*hint*
The sentence parts are:
The foolish soldier / doesn't seem / happy.
Try again.
>

ILIAD's knowledge of the sentence structure allows it to show the subject, verb and its auxiliary, and predicate adjective separately no matter how complex those parts are.

The Question Tutorial teaches the student how to form different types of questions from the corresponding declarative forms. In the current version of this tutorial the student may elect to form yes-no questions, questions on the subject of the sentence, or questions on the object. (ILIAD actually has the ability to generate a wider variety of question forms than just these three.)

In the following example, the user has instructed the system that s/he wishes to make yes-no questions at a medium complexity level with a high level vocabulary. ILIAD generates an appropriate declarative sentence and its corresponding question form, but displays only the former to the student. The system checks the student's input[3] against the interrogative form to determine whether the student has produced the correct response.

Claire didn't unlock the gate.
Make a yes-no question.
>*Didn't Claire unlock the gate?*
That's right.

In the following interaction the student has selected a relatively difficult comprehension task, questions on the object in complex sentences.

Bette was shown the book by the adult who helped Mrs. Rowe.
Make a Wh-question about the object.
>*What was Bette shown by the adult who Mrs. Rowe helped?*
Right!!

The generative mechanism that produces sentences in such a way that the system understands their structure is not limited to application in tutorials that focus on basic syntactic phenomena. ILIAD is also capable of integrating such information with knowledge of how language is used in various concrete situations to instruct the student in the *functional use* of language. One such function of language is to make requests.

The Request Tutorial facilitates the student's ability to use request forms in two

---

[3]This tutorial is not programmed to insist on correct punctuation or capitalization, although this could in principle be done; see the second exercise below for an example.

ways. It introduces the student to the relative politeness of different types of requests, ranging from stark imperatives to longer, more formal and gracious forms. It also teaches the student about the correct syntactic form of requests, such as which request types may appear with "please" and where "please" may appear in those requests. Here are two examples of this tutorial:

The kid wanted Pat to fix the bikes.
He said, **"How about fixing the bikes?"**
Was he being BOSSY, STRONG, POLITE, or VERY POLITE?
>POLITE
That was very close, Bob.
The kid was not being POLITE.
He was being more direct.
Try again.
>STRONG
Terrific!

In the next example, the user is presented with a request form which may be made more polite by adding the word PLEASE. The user must insert this word in all the places where it is co rect to do so.

Bake the gentlemen a cake!
>I would appreciate it if you would bake the gentlemen a cake, please.
There are 2 more ways to add PLEASE.
>I would appreciate it if you would please bake the gentlemen a cake.
There is 1 more way to add PLEASE.
Please, I would appreciate it if you would bake the gentlemen a cake.
Perfect!

At present we are working on an implementation of ILIAD that is designed to run using two microcomputers. One computer, the MicroILIAD, will generate the sentences, while the other computer, the MicroTUTOR will be used to generate animated sequences that correspond to the meaning of those sentences and interact with the student.

4. Conclusion

The generative approach is extremely powerful and efficient as a design base for instructional systems. Expanding one element of the generative system - such as the vocabulary or the rules for generating sentences - exponentially increases the number, range, and/or complexity of sentences that can be created and thus automatically increases the power of existing tutorials. It is flexible in that different tutorials are all based on the same generation mechanism. Thus many varieties of tutorials are possible. Table 1 summarizes the differences between traditional and generative systems for language instruction.

Going beyond grammar instruction, the generative approach may also be used to address more contextual aspects of language use. A tutorial on appropriate use of request forms might focus the student's attention on the importance of the relative

social status of speakers and hearers. Tutorials on inference would improve understanding of both dialogue and written text by creating both appropriate and inappropriate inferences to test this broader form of comprehension. To fully exploit this approach, further research will be needed to develop systems capable of creating paragraphs of text and stories tailor-made to student language learning needs and comprehension ability.

The generative approach is not limited to language instruction; any instructional domain that can have a set of rules for generating material is a suitable candidate. One possible area is chemistry, where an instruction system could create equations to be balanced; another is physics, where problems in mechanics or other areas could be rule-generated. Of course many types of mathematics could also be taught this way.

As computer power becomes available to large numbers of users at a very low cost, we must be prepared to take advantage of this exciting resource with similarly powerful software. The concept of a generative, knowledge-based system for education inspires the development of that software.[4]

TABLE 1: Approaches to Computer Assisted Language Instruction

[4]We would like to acknowledge the work of the other members of the ILIAD project: Dianne Brown, Varda Shaked and William Simpson. The work reported here was sponsored by the Office of Special Education, Department of Education, Grant # G007803230.

Traditional	Generative
Fixed number of examples	Limitless examples
Extensive storage required	Compact
Teaches isolated skills	Teaches combinations of skills
Little user control	User controls vocabulary and syntactic complexity
Inadequate remedial resources	Cannot be exhausted
Hard to add new words	Easy to increase vocabulary
Often not theoretically based	Based on general linguistic formalism
Presents pre-stored information	Generates information from rules
Not knowledge-based	Uses knowledge of material
Limited capacity for improvement	Easy to add new capabilities
Hard to tailor to special needs	Can generate sentences typical of problem populations

I. APPENDIX 1 -- Sample Base and Surface Structures

II. APPENDIX 2 -- Sample Sentences Generated by ILIAD

The baby cried.
The baby was crying.
The baby hadn't been crying.
The babies are crying.
Was the baby crying?
Did the baby cry?
Who cried?
Who was crying?
Who had been doing what?
What did the baby do?
What had the baby been doing?
He was crying.
The baby didn't cry.

Pat gave Bette a doll.
Pat gave a doll to Bette.
Bette was given a doll by Pat.
Bette was given a doll.
A doll was given to Bette.
Who was given a doll by Pat?
What was Bette given?
Who was Bette given a doll by?
Whom was Bette given a doll by?
By whom was Bette given a doll?
Who gave what to whom?

There is a tricycle on the grass.
The fly didn't fly.
The kid was angrier than Kirk.
Kevin is a foolish kid.
The cake looked good.
Some hungry people will eat.
Not every man will play the game.
The doctor was depressed.
A girl wanted Bob to send post cards to the kid.
The shirt got torn.
The post card got stolen.
What does the man who Bob tripped read?
What did the ladies who tickled Judy wear?
Alicia is hungrier than Jake.
The excited children don't know whether the hungry girls will swear.
The sad woman will cry.
Varda won't kiss the boys.
Some hungry people eat.
Every new toy broke.

The boy won't give the dogs any oranges.
The girls were hard to ridicule.
The girls were hard for Varda to ricicule.
The children were angrier than Andy.
The smartest bully bought the bananas.
The sad doctor must have kidnapped the girl.
The teacher ought to have sent Nancy a letter.
Ms. Brown should have made Kirk a cake.
The men would have been told new rules by the teacher.
Kirk was able to read the book.
Four girls were on the slide.
Should Dianne have given Pat a baseball?
Are you going to unlock the gate or not?
Can you send the post card?
Don't you think you should send Cathy letters?
Did David get hugged?
Must John leave?
Which movies do you like?
Bob knew who ran.
Who did Bob promise to help?
The teacher who the boy kissed read the book.
Lyn has the sweet cookie which the woman baked.
The teacher who answered Nancy was foolish.
The young boy who forgot the rules made up the answers.
You had better read the stupid men the books.
Lyn knows the people who they watched.

You open the door!
Open the door!
Open the door, please!
You had better open the door.
You will open the door!
You'd better open the door.
Don't you think you should open the door?
Are you going to open the door or not?
Can you open the door?
Please, can you open the door?
Will you open the door?
Will you please open the door?
Could you open the door?
Please, could you open the door?
Would you open the door, please?
I want you to open the door.
Want to open the door?
I think you should open the door.
Why don't you open the door?
Do you want to open the door?
Do you intend to open the door?
Is it possible for you to open the door?
You will open the door, won't you?

Why not open the door?
Are you able to open the door?
Open the door, won't you?
I would like you to open the door.
How about opening the door?
How's about opening the door?
What do you say you open the door?
How would you like to open the door?
It would be nice if you opened the door.
It would be nice if you would open the door.
Would you mind opening the door?
Do you think you could please open the door, please?
It would be nice if the door were opened.
I would appreciate it if you would open the door.
Would it be possible for you to open the door?
Wouldn't it be nice if someone were to open the door?
It would be nice if the door were opened.
I would like it if the door were opened.
Wouldn't it be nice if the door were opened?

References

[1]

Allee, John G., Peter J. Quinn and Robert L. Williams.
An Elementary School Language Arts CAI Series.
The George Washington University, Washington, D.C., 1977

[2]

Bates, Madeleine and Kirk Wilson.
A Generative Computer System to Teach Language to the Deaf.
In *Proc. ADCIS Conference*. Proc. 1979 ADCIS Conf., February, 1979.

[3]

Brown, John Seely, R. R. Burton, and A. G. Bell.
SOPHIE: A Step Towards A Reactive Learning Environment.
International Journal of Man Machine Studies 7:675-696, 1975.

[4]

Brown, John Seely.
Uses of Artificial Intelligence and Advanced Computer Technology in Education.
In *Computers and Communications* . Academic Press, New York, 1977.

[5]

Burton, Richard R. and John Seely Brown.
Intelligent CAI: An Author Aid for a Natural Language Interface.
Academic Press, New York, 1979, .

[6]

Carroll, John B.
Towards a Performance Grammar of Language Production.
University of North Carolina, Chapel Hill, North Carolina, 1978

[7]

Findler, Nicholas V. (Ed.).
Associative Networks: Representation and Use of Knowledge by Computers.
Academic Press, New York, 1979.

[8]

Helm, Carl, E. and Charlotte McIver.
Automated Sentence Analysis for Language Instruction.
Computers and the Humanities 8:239-245, 1974.

[9]

Kehler, T.P.
Teaching English Grammar Using A Microcomputer.
Texas Women's University, Denton, Texas, 1977

[10]

O'Neil, Harold F. (Ed.).
Instructional System Development.
Academic Press, New York, 1979.

[11]

Seidel and Rubin (Eds.).
Computers and Communications.
Academic Press, New York, 1977.

[12]

Weischedel, R.M., W.Voge and M. James.
An Artificial Intelligence Approach to Language Instruction.
Artificial Intelligence 10:225-240, 1978.

[13]

Weischedel, R.M. and J. Black.
Responding to Potentially Unparseable Sentences.
Journal of the Association for Computational Linguistics 6:2:97-109, 1980.

[14]

Williams, Robert L., Peter J. Quinn and John G. Allee.
A CAI Approach to the Development of English Skills: English Usage Exercises I.
St. John's University, Jamaica, New York, 1977

[15]

Williams, Robert L.
Sentence Generator.
St. John's University, Jamaica, New York, 1977

[16]

Williams, Robert L.
Sentence Construction With a Computer.
Creative Computing :52-58, April, 1980.

Table of Contents

1. Introduction	0
2. The Power of a Generative System	1
3. Examples of Knowledge Use	3
4. Conclusion	7
I. APPENDIX 1 -- Sample Base and Surface Structures	10
II. APPENDIX 2 -- Sample Sentences Generated by ILIAD	11
References	14